AN EXAMPLE OF COMPUTER PROGRAMMING

The following problem was considered for publication in "Parabola", but rejected as the editorial committee could find no reasonable way of solving it. "The number 273 (written to the base 10) is equal to 333 (written to the base 9) and also to 111 (written to the base 16). What other integers $M \leq 10000$ can be represented as KKK (written to the base $J > K$) in two different ways"?

This problem is very suitable for solution by computer. So let's work out a way of solving it. Suppose you had a supply of paper with 10000 lines numbered 1, 2, ... , 10000 with the numeral 0 on each line. Note that the only possible choices of base J are 2, 3, 4, ... , 99 (for $100^2+100+1 > 10000$). For each of these values, you could calculate J^2+J+1. Then the integer K must be at least 1, and also

$$K \leq J-1$$
$$(J^2+J+1)K \leq 10000$$

i.e. $K \leq$ minimum of $J-1$ and $10000/(J^2+J+1)$.

So you could, for each value of K from 1 to this upper bound, calculate $K(J^2+J+1) = M$, cross out the number on line M and write in the next higher number. This could be done for each value of J. Then the list could be searched for numerals greater than 1.

This would be very tiresome to be done by hand, but electronically the process is reasonably efficient. So these ideas were the basis of the following program in the language FORTRAN.

```
0001        DIMENSION NUMBER(10000)
0002        DO 100 I = 1, 10000
0003    100 NUMBER(I) = 0
0004        DO 200 J = 2, 99
0005        JX = J*J + J + 1
0006        JMIN = MIN0(J-1,(10000/JX))
0007        DO 200 K = 1, JMIN
0008        M = K*JX
0009    200 NUMBER(M) = NUMBER(M) + 1
0010    300 FORMAT (20X, I5, ' HAS ', I1, ' SUCH REPRESENTATIONS.'))
0011        DO 400 I = 1, 10000
0012        IF (NUMBER(I).LT.2) GO TO 400
0013        WRITE (3,300) I, NUMBER(I)
0014    400 CONTINUE
0015        STOP
0016        END
```

Let's go through it line by line.

0001 This tells the machine to choose 10000 places in its memory, and call them NUMBER(1), ... , NUMBER(10000).

0002 This is a simple DO-loop. The machine is told to work its way down to the line labelled 100 (the programmer labels lines only when necessary) using the value I = 1, then again with the value I = 2, and so on - 10000 times in all.

0003 This tells the machine to cancel whatever was stored in memory place NUMBER(I), and put the number 0 there instead.

0004 This starts another DO-loop. It is clear that the possible choice of bases J are between 2 and 99 (note $100^2+100+1 > 10000$).

0005 The integers representable as KKK (written to the base J) are the multiples of J^2+J+1. So the machine is told to calculate this number and to put it in a phase in its memory called JX. These names of places are chosen to help the programmer.

0006 The possible choices of the integer K are limited by the requirements that it be at least 1 and at most J-1. We also need the product K*JX at most 10000. So the machine calculates (using the standard function MIN0) the lesser of J-1 and 10000/JX. The machine rounds off 10000/JX to the integer below. It is stored in a new place, JMIN.

0007 Here a third DO-loop (nested inside the second) starts. We will consider successively values of K from 1 to JMIN.

0008 The integer M = J*JX is calculated. It is at most 10000 (remember how we defined JMIN?)

0009 The integer stored in place NUMBER(M) is now increased by 1. So NUMBER(I) will ultimately contain the number of such representations of I. Both the second and third DO-loops terminate here.

0010 This just prescribes a method for writing out two integers, one of five digits, one of one digit. As it will be used later, it is labelled 300 for reference. This line could be put elsewhere in the program.

0011 This starts another DO-loop.

0012 The machine is told to determine whether the integer in place NUMBER(I) is less than 2. If it is, it skips line 0013 and goes to labelled line 0014. As we will see, it does nothing for these values of I.

0013 This line is reached only for those values of I which have two such representations. I, and NUMBER(I) are written out by the method labelled 300.

0014 This is a dummy line to mark the end of the fourth DO-loop. Why would it have been wrong to omit this and label the previous line 400?

0015 The machine is told to stop when it reaches this line.

0016 END shows that the program has ended.

The final printout looked like this:

```
 273 HAS 2 SUCH REPRESENTATIONS.
 546 HAS 2 SUCH REPRESENTATIONS.
 931 HAS 2 SUCH REPRESENTATIONS.
3549 HAS 2 SUCH REPRESENTATIONS.
3783 HAS 2 SUCH REPRESENTATIONS.
4557 HAS 2 SUCH REPRESENTATIONS.
7566 HAS 2 SUCH REPRESENTATIONS.
9114 HAS 2 SUCH REPRESENTATIONS.
```

You may care to find these representations.

The program can easily be changed to examine the integers ≤ 100000. The new output lists twenty-six numbers between 10000 and 100000 with two such representations and one (67053) with three such representations.

$$\begin{aligned} 67053 &= 3(149^2 + 149 + 1) \\ &= 21(56^2 + 56 + 1) \\ &= 31(46^2 + 46 + 1). \end{aligned}$$

Peter Donovan.

Note: If you have a comparable problem which you would like run on the computer, write in and we will see what we can do - Ed.

* * *

Try this card trick

Cut a pack of cards into two piles so that one pile contains between 20 and 30 cards. (This is essential for the trick to work.) Count the number of cards in the smaller pile and add the digits of the number together. For example, if there were 23 cards in the smaller pile you would add 2 and 3 to get 5. Now look at the fifth card from the bottom of the smaller pile, replace it, and put the smaller pile on top of the larger pile. The card you looked at will be the 19th card from the top. (Explanation on page 36)